Jean Saquet – Université de Caen

Master d'Informatique

Réseaux avancés

Proxies et filtrage applicatif

Bureau S3-354
Mailto:Jean.Saquet@unicaen.fr
http://saquet.users.greyc.fr/M2/rezo

Proxy applicatif

Un proxy, ou serveur mandataire, relaie une requête venant d'un client à destination d'un serveur non directement accessible en général.
Contrairement à la passerelle NAPT qui agit au niveau transport (couches réseau et transport), le proxy agit au niveau application (http, ftp, ...etc).
Différence : la passerelle NAPT relaie les datagrammes IP, le proxy relaie les messages applicatifs.

Proxy et transition v4/v6

Un relais applicatif (http, smtp et autres protocoles de mails, DNS, ...) peut être très utile pour faire communiquer un monde v4 et un monde v6 (aussi bien que monde privé / monde public).

En effet, requêtes et réponses du niveau applicatifs sont alors reçues dans un des protocoles IP et renvoyés avec l'autre, dans des datagrammes différents.
Donc pas besoin de mécanismes complèxes de traduction de datagrammes.

Proxy : utilité

• Accès à certaines applications de l'Internet à l'intérieur d'un réseau privé.
Exemple typique : proxy http. Intègre généralement un proxy DNS pour que le client puisse résoudre les adresses.
• Mémoire cache des pages demandées. Evite des accès ultérieurs pour les sites les plus consultés
• Possibilité de filtrage au niveau applicatif (par le contenu par exemple).

Serveurs internes

Un serveur peut être placé dans un domaine privé (avec des adresses privées v4, ou des règles de filtrage en interdisant l'accès direct).
Intérêt : meilleur contrôle des accès à ce serveur et à ceux qu'il utilise (base de données, ...).

On peut alors utiliser un NAT (avec règle de translation fixe) pour l'accès à partir de l'extérieur.
Mais : le NAT agit au niveau réseau/transport.

Reverse-proxy

Afin d'agir au niveau applicatif, on peut utiliser un reverse-proxy pour permettre l'accès à un (ou plusieurs) serveur(s) interne(s).

Le principe de base est simple : la requête émise par le client extérieur est renvoyée par le proxy au serveur compétent pour y répondre.

Avantages (et inconvénients ...) sont toutefois multiples et plus complexes qu'avec un NAT.

Reverse-proxy, avantages

- Répartion des requêtes sur plusieurs serveurs, par exemple en fonction des langages utilisés (Perl, PHP, différentes versions de PHP, ...)
- Contrôle / répartition de charge
- Virtual hosts (regroupement de serveurs sur une même IP)
- Simplification des règles firewall
- Utilisation du cache
- Frontal pour certaines applications (Zope, ...)
- Filtrage applicatif (le principal intérêt)

Reverse-proxy, inconvénients

- Point central, pb majeur en cas de panne.
- Serveurs multiples (coût), sauf virtualisation, et règles de ré-écriture parfois complexes.
- Contrôle par IP du client impossible au niveau de chaque serveur (ne voient que celle du proxy), par contre globalisation au niveau du proxy.
- Rupture SSL en cas d'utilisation de https ("man in the middle").Ce peut être un avantage.

Reverse-proxy, logiciels

Qualités souhaitables : virtual hosts, ssl, cache, ré-écriture d'URLs.

Solution commerciale de Microsoft : Internet Security and Acceleration Server
Monde libre : Squid, Apache ou quelques solutions plus spécialisées.

Modules additionnels pour Apache (exemple : mod_security).

Http : attaques

Possibilté de faire exécuter des commandes (p.e. Mv, chmod, ...), en utilisant des failles de logiciels. (scripts Perl, Php, ...)

Ces attaques ont le plus souvent des formats connus.
==> filtrage au niveau applicatif par règles analysant les requêtes.

Filtrage applicatif (p.e. http)

L'intérêt du reverse-proxy est de regrouper toutes les règles de filtrage et de redirection au niveau applicatif.

Reverse-proxy et filtrage applicatif constituent donc une architecture intéressante.

Exemple : module mod-security, associé à Apache utilisé en reverse-proxy .

Filtrage avec mod-security

5 phases d'intervention possibles :
Analyse de l'en-tête, puis du corps de la requête.
Analyse de l'en-tête, puis du corps de la réponse.
Journalisation de l'opération.

Dans la règle, on peut préciser la phase d'application. Il y a bien sûr des limitations : on ne peut pas appliquer une règle en phase 1 si elle nécessite de connaître le corps de la requête, par exemple.

Mod_security - Limitations

Les attaques fonctionnant par effet de bord doivent être détectées en phase 1 pour ne pas atteindre le serveur

Exemple : injection SQL dans une URL
Des variables sont utilisées dans une requête SQL, on y ajoute des compléments
Ex 'OR 1=1' ...

Règles de mod-security

Forme générale :

Secrule VARIABLES OPERATEUR [ACTION]

- Variables à tester
- Opérateur pour utilisation d'expression régulière
- Action (optionnelle car peut être générale) pass, deny, drop, ... permet également de spécifier la phase et le fait de journaliser ou non.

Mod-security : activation

Le fichier de configuration Apache doit préciser qu'on utilise le module mod_security,
et également des fichiers indispensables :
La library libxml2.so
Le langage lua liblua.so.

Le fichier Modsecurity.conf permet d'activer le mode, de définir des règles générales ou particulières, et d'inclure des fichiers de configurations "standards" (ces derniers pouvant être modifiés)

Mod-security : qqs règles générales

- Analyse des corps des requêtes et réponses
- Traitement des logs par modules spécialisés
- Limites de taille de messages
- Scan des variables POST
- …

Mod-security : les includes

Fichiers de règles "standard" pour des attaques connues :

- bloquer l'accès aux commandes sensibles
- test de violation du protocole http
- blocage d'extensions de fichiers, accès bd
- détection de faux robots
- détection des chevaux de Troie connus
- SQL injection
- ...cf. TP

Mod-security : défauts

Consommation mémoire et Cpu : à surveiller
Risque de blocage de certaines applications
Nécessiter de personnaliser les règles
Risque d'interdire certains accès pourtant souhaités